AD-A059 968 STANFORD UNIV CALIF DIGITAL SYSTEMS LAB F/G 9/2
DIRECTLY EXECUTED LANGUAGES.(U)
SEP 78 M J FLYNN DAAG29-76-G-0001
UNCLASSIFIED ARO-12958.11-EL NL

1 OF 1
ADA
059968

END
DATE
FILMED
12-78
DDC

(1)

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(12) **LEVEL** II

(18) ARO

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER (19) 12958.11-EL	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER (9)	
4. TITLE (and Subtitle) (6) Directly Executed Languages	5. TYPE OF REPORT & PERIOD COVERED Final Report 15 Jul 75 - 14 Jul 78
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Michael J. Flynn	8. CONTRACT OR GRANT NUMBER(s) (15) DAAG29-76-G-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS Stanford University Stanford, California 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709	12. REPORT DATE (11) Sep 78
	13. NUMBER OF PAGES 6
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) (12) 8p.	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

DDC RECEIVED OCT 16 1978 B

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

78 10 10 068

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The research addressed two important issues in the design of optimal languages for direct execution in an interpretive system: binding the operand identifiers in an executable instruction unit to the arguments of the routine implementing the operator defined by that instruction; and binding operand identifiers in an executable instruction unit to the arguments of the routine implementing the operator defined by that instruction; and binding operand identifiers to execution variables. These issues are central to the performance of a system, both in space and time. This report summarizes the results of this study.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

408 071

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DIRECTLY EXECUTED LANGUAGES

Final Report

Prepared For

Department of the Army
U.S. Army Research Office

Contract No. DAAG29-76-G-0001

July 15, 1975 - July 14, 1978

by

Michael J. Flynn
Digital Systems Laboratory
Department of Electrical Engineering
Stanford University
Stanford, CA 94305

Statement of the Problem

Representation is a key problem in computer systems. This manifests itself in many independent ways: the accuracy, the validity, the efficiency in human productivity, problems of design representation in the creation of new systems, etc. The research undertaken in this project involves one aspect of the representation issue; the production of highly efficient program representations for machine execution. This representation corresponds to a machine language in that it represents the commands which are interpreted by a machine. However, unlike conventional machine language approaches the representation is tailored to particular higher level language environments. The problem then is to find ways of synthesizing such very efficient language representations. We call languages thus derived, Directly Executed Languages or DELs.

Research Summary

A computer is largely defined by its instruction set. Of course, other issues such as space, power, algorithms used, may be important in certain applications but the user basically sees the instruction set of the machine. The instruction set, thus, is the interface between programs and resources. The program is a sequence of instructions that accomplish a desired user end. The instructions are interpreted by a control unit which activates the system's resources (data paths) to cause proper transformations to occur.

The instruction set is referred to as the architecture of the processor. It is actually a language whose usefulness is best measured by the space it requires to represent a program and time required to interpret these representations. Recent developments in technology allow a great deal more flexibility in control unit structure while a variety of current research efforts have brought additional understanding in the nature of the instruction set. The purpose of our research was to explore these developments, especially the relationship between an arbitrary higher level language program representation and an "ideal" architecture for that language.

Specifically our research addressed two important issues in the design of optimal languages for direct execution in an interpretive system: binding the operand identifiers in an executable instruction unit to the arguments of the routine implementing the operator defined by that instruction; and binding operand identifiers in an executable instruction unit to the arguments of the routine implementing the operator defined by that instruction; and binding operand identifiers to execution variables. These issues are central to the performance of a system, both in space and time.

Historically, some form of "machine language" is used as the directly executable medium for a computing system. These languages traditionally are constratined to a single "n-address" instruction format; this leads to an excessive number of "overhead" instructions that do nothing but move values from one storage resource to another being imbedded in the executable instruction stream. We have developed techniques to reduce this overhead by increasing the number of instruction formats available at the directly executed language level [10].

Machine languages are also constricted with respect to the manner in which operands can be "addressed" within an instruction. Usually, some form of indexed base-register scheme is available, along with a direct addressing mechanism for a few, "special" storage cells (i.e., registers, and perhaps the zeroth page of main store). We developed a different identification mechanism--based on the Contour Model of Johnston. Using our scheme, only N bits are needed to encode any identifiers in a scope containing less than 2**N distinct identifiers.

Together, these two results lead to directly executed language designs which are optimal in the sense that: (1) k executable instructions are required to implement a source statement containing k functional operators; (2) the space required to represent the executable form of a source statement containing k distinct funcitonal operators and v distinct variables approaches F*k + N*v -- where there are less than 2**F distinct functional operators in the scope of definition for the source statement, and less than 2**N distinct variables in this scope; (3) the time needed to execute the representation of a source statement containing k functional operators, d distinct variables in its domain, and r distinct variables in its range approaches d + r + k; where time is measured in memory references.

In order to test the above results a novel directly executed language (DELtran) [9] tailored specifically to the FORTRAN source language, EMMY host, and scientific programming was constructed. DELtran is "transformationally complete" in that:

(1) Code generation is linear with respect to the number of operators in a FORTRAN program.

(2) Only k DELtran instruction units are needed to represent a FORTRAN statement containing k functional operators.

(3) The space needed to represent a FORTRAN statement approaches N*v+F*k-- where v is the number of distinct variables in the statement, and N and F are the least integers such that there are less than 2**N distinct variables and 2**F distinct operators in the relevant scope of definition.

In addition, DELtran is "transparent" in that there is a 1-1 correspondence between DELtran operators and control constructs and FORTRAN operators and control constructs, and "invertible" in that all sensible sequences of DELtran instruction units have a direct FORTRAN analogue.

The performance and vital statistics of DELtran on the host EMMY [8] is interesting, especially when compared to the 370 performance on the same system. The table below is constructed using a version of the well-known Whetstone benchmark and widely accepted and used for FORTRAN machine evaluation. The EMMY host system referred to in the table is a very small system--the processor consists of one board with 305 circuit modules and 4096 32 bit words of interpretive storage. It is clear that the DELtran performance is significantly superior to the 370 in every measure.

DELtran vs. System 370 Comparison for the Whetstone Benchmark

```
Whetstone Source -- 80 statements (static)
                 -- 15,233 statements (dynamic)
                 -- 8,624 bits (excluding comments)
```

	System 370 FORTRAN-IV opt 2	DELtran	ratio 370/Deltran
Program Size (static)	12,944 bits	2,428 bits	5.3:1
Instruction Executed	101,016 i.u.	21,843 i.u.	4.6:1
Instruction/Statment	6.6	1.4	4.6:1
Memory References	220,561 ref.	46,939 ref.	4.7:1
EMMY Execution Time (370 emulation approximates 360 Model 50)	0.70 sec.	0.14 sec.	5:1
Interpreter Size (excludes I/O)	2,100 words	800 words	2.6:1

Publications

[1] Hedges, Thomas S., "EMMY/360 Cross Assembler", Digital Systems Laboratory TN 74, Stanford University, Stanford, CA, December 1975.

[2] Wallach, Walter A., "EMMYXL User's Guide", Digital Systems Laboratory TN 84, Stanford University, Stanford, CA, March 1976.

[3] Polstra, John D., "EMMYPL User's Manual", Digital Systems Laboratory TN 86, Stanford University, Stanford, CA, April 1976.

[4] Wallach, Walter A., "EMMY/UNIBUS Interface, Preliminary Specification", Digital Systems Laboratory TN 88, Stanford University, Stanford, CA, June 1976.

[5] Wallach, Walter A., "Virtual Addressing for the EMMY/360", Digital Systems Laboratory TN 89, Stanford University, Stanford, CA, June 1976.

[6] Wallach, Walter A., "EMMY/360 Functional Characteristics", Digital Systems Laboratory TR 114, Stanford University, Stanford, CA, June 1976.

[7] Hoevel, Lee and Wallach, Walter, "Emulation Oriented Software First Development", Digital Systems Laboratory TN 95, Stanford University, Stanford, CA, August 1976.

[8] Flynn, M. J., "The Interpretive Interface: Resources & Program Representation in Computer Organization", Proceedings of the Symposium on High Speed Computers and Algorithm Organization (Academic Press), April 13-15, 1977, University of Illinois, Champaign, Ill., pp. 41-69.

[9] Hoevel, Lee, "DELtran Principles of Operation: A Directly Executed Language for FORTRAN-II", Digital Systems Laboratory TN 108, Stanford University, Stanford, CA, March 1977.

[10] Hoevel, L. W. and Flynn, M. J., "The Structure of Directly Executed Languages: A New Theory of Interpretive System Design", Digital Systems Laboratory TR 130, Stanford University, Stanford, CA , March 1977.

[11] Flynn, M. J., "Computer Organization and Architecture", Lecture Notes on Advanced Operating Systems, 1978 (Pub. Springer-Verlag), pp. 17-98.

[12] Wallach, Walter A., "EMMY/360: An Emulation of System/360 for the Stanford EMMY", The 11th Symposium on Microprocessing, Sponsored by the ACM, IEEE Computer Society, November 19-22, 1978.

Scientific Personnel

Michael J. Flynn
Professor, Electrical Engineering
Stanford University

Lee W. Hoevel
(Received Ph.D. degree in Electrical Engineering from Johns Hopkins University, June 1978)

Scott Wakefield
(to receive Ph.D. degree in Electrical Engineering June 1979)

Walter A. Wallach
(to receive degree in Electrical Engineering June 1979)